

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

1 This application is submitted in the name of the following inventor(s):

2
3 Inventor

Citizenship

Residence Address

4 MALCOLM, Michael

United States

Los Altos, California

5
6 The assignee is CacheFlow, Inc., a Delaware corporation having an office at 650 Almanor
7 Avenue, Sunnyvale, CA 94086.

8
9 Title of Invention

10
11 Multiple Cache Communication

12
13 Background of the Invention

14
15 This application is a continuation of application number 09/127,249 filed 7/31/98, and of PCT
16 application Serial Number PCT/US99/17149 filed July 28, 1999, hereby incorporated by reference.

17
18 1. *Field of Invention*

19
20 The invention relates to caches.

21 2. *Related Art*

22
23 In a computer system in which client devices request information from one or more server
24 devices, it is sometimes desirable to provide cache; that is, a device that

1 maintains copies of requested information so multiple requests for the same information
2 can be satisfied at the cache. When requests for information are satisfied at the cache, the
3 server devices need not receive the requests, process them, and retransmit the same in-
4 formation over a communication channel that links the client devices and the server de-
5 vices. For example, the server devices can be web servers, the client devices can be web
6 clients, the communication channel can be an IP network such as the Internet, and the re-
7 quested information can be web objects.

8
9 Some information requested from the server devices is considered not
10 cacheable, for one or more of several reasons. As examples, the server can refuse to allow
11 the information to be cached, or the information can be a result of a dynamic process that
12 can provide differing results for the same request (so caching would obviate the operation
13 of that dynamic process). An example of dynamically processed information could in-
14 clude advertisements, database searches, or output from CGI scripts.

15
16 However, it often occurs that non-cacheable information is requested a sec-
17 ond time without having changed, so the second request to the server results in identical
18 information being returned. In a system with multiple communicating caches, transmit-
19 ting the same information from a first cache to a second cache (when each already has a
20 copy) is an inefficient use of communication resources.

9

10

11

12

1 receiving cache can reliably identify the compressed information in response to the mes-
2 sage, and compression in which the receiving cache will sometimes be unable to identify
3 the compressed information.

4
5 In a third aspect of the invention, a first cache refrains from unnecessarily
6 transmitting the same information to a second cache when each already has a copy. This
7 includes both maintaining a record at a first cache of information likely to be stored at a
8 second cache, and transmitting a relatively short identifier for that information in place of
9 the information itself.

10
11 In a preferred embodiment, a set of caches are disposed in a directed graph
12 structure, with a set of root caches disposed for coupling to server devices and a set of
13 leaf caches disposed for coupling to client devices. Both root caches and leaf caches
14 store non-cacheable objects beyond their initial use, along with relatively short identifiers
15 for the non-cacheable objects. When a server device returns identical information to a
16 root cache in response to a request for a non-cacheable object, that root cache transmits
17 only the identifier of the non-cacheable object to the requesting leaf cache, avoiding re-
18 transmitting the entire object if the leaf cache still has the object.

19 20 Brief Description of the Drawings

21
22 Figure 1 shows a block diagram of a system having multiple caches.

Figure 2 shows a process flow diagram for a method of using a system having multiple caches.

Detailed Description of the Preferred Embodiment

In the following description, a preferred embodiment of the invention is described with regard to preferred process steps and data structures. Those skilled in the art would recognize after perusal of this application that embodiments of the invention can be implemented using one or more general purpose processors or special purpose processors or other circuits adapted to particular process steps and data structures described herein, and that implementation of the process steps and data structures described herein would not require undue experimentation or further invention.

Inventions disclosed herein can be used in conjunction with inventions disclosed in one or more of the following patent applications:

o Provisional U.S. Application 60/048,986, filed June 9, 1997, in the name of inventors Michael Malcolm and Robert Zarnke, titled "Network Object Cache Engine", assigned to CacheFlow, Inc., attorney docket number CASH-001 (P).

1 o U.S. Application Serial No. 08/959,058, filed October 28, 1997, in the name of in-
2 ventors Michael Malcolm and Ian Telford, titled "Adaptive Active Cache Re-
3 fresh", assigned to CacheFlow, Inc., attorney docket number CASH-003.

4
5 o U.S. Application Serial No. 08/959,313, filed October 28, 1997, in the name of in-
6 ventors Doug Crow, Bert Bonkowski, Harold Czegledi, and Tim Jenks, titled
7 "Shared Cache Parsing and Pre-fetch", assigned to CacheFlow, Inc., attorney
8 docket number CASH-004.

9
10 o U.S. Application Serial No. 09/093,533, filed June 8, 1998, in the name of inven-
11 tors Michael Malcolm and Robert Zarnke, titled "Network Object Cache Engine",
12 assigned to CacheFlow, Inc., attorney docket number CASH-001.

13
14 and

15 o PCT International Application PCT/US 98/11834, filed June 9, 1997, in the name
16 of assignee CacheFlow, Inc., and inventors Michael Malcolm and Robert Zarnke,
17 titled "Network Object Cache Engine", attorney docket number CASH-001 PCT.

18
19 These applications are referred to herein as the "Cache Disclosures," and
20 are hereby incorporated by reference as if fully set forth herein.

21
22 / / /

System Elements

Figure 1 shows a block diagram of a system having multiple caches.

A system 100 includes a cache system 110, at least one client device 120, and at least one server device 130.

Client Device

Each client device 120 is coupled to the cache system 110 using a client communication path 121. The client communication path 121 can include a dial-up connection, a LAN (local area network), a WAN (wide area network), an ATM network, an IP network (such as an internet, intranet, or extranet), or some combination thereof. In a preferred embodiment, the client communication path 121 includes a dial-up connection, such as for coupling a subscriber to an ISP (internet service provider), or a LAN, such as for coupling a workstation to an internet connection.

As used herein, the terms "client" and "server" refer to relationships between the client or server and the cache 110, not necessarily to particular physical devices.

1 As used herein, the term "client device" includes any device taking on the
2 role of a client in a client-server environment. There is no particular requirement that the
3 client devices 110 must be individual devices; they can each be a single device, a set of
4 cooperating devices, a portion of a device, or some combination thereof.

5
6 Server Device

7
8 Each server device 130 is also coupled to the cache system 110 using a
9 server communication path 131. The server communication path 131 can include a dial-
10 up connection, a LAN (local area network), a WAN (wide area network), an ATM net-
11 work, an IP network (such as an internet, intranet, or extranet), or some combination
12 thereof. In a preferred embodiment, the server communication path 131 includes an
13 internet backbone and an internet connection between the cache system 110 and the inter-
14 net backbone.

15
16 As used herein, the term "server device" includes any device taking on the
17 role of a server in a client-server environment. There is no particular requirement that the
18 server devices 130 must be individual devices; they can each be a single device, a set of
19 cooperating devices, a portion of a device, or some combination thereof.

20

1 The server device 130 includes memory or storage 132 for recording one or
2 more web objects 133. The web objects 133 can include any type of data suitable for
3 transmitting to the client device 110, such as the following:

4
5 o text, color, formatting and directions for display;

6
7 o pictures, data in graphical formats (such as GIF or JPEG), other multimedia data;

8
9 o animation, audio (such as streaming audio), movies, and video (such as streaming
10 video), and other data in audio or visual formats (such as MPEG);

11
12 o program fragments, including applets, Java, JavaScript, and ActiveX; and

13
14 o other web documents (such as when using frames);

15
16 and

17 o other data types (such as indicated by future extensions to HTML, DHTML,
18 SGML, XML, or similar languages).

19
20 / / /

1 Cache System

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

The cache system 110 includes a set of caches 111. Each cache 111 is designated a "leaf cache" if it is coupled to one or more client communication paths 121, and is designated a "root cache" if it is coupled to one or more server communication paths 131. The cache system 110 includes an inter-cache communication path 112 for communication between and among caches 111.

The inter-cache communication path 112 can include a plurality of direct connections, a LAN (local area network), a WAN (wide area network), an IP network (such as an internet), or some combination thereof. In a preferred embodiment, the inter-cache communication path 112 includes a plurality of direct connections between pairs of individual caches 111.

In a preferred embodiment, the caches 111 in the cache system 110 are disposed in a graph structure. One or more leaf caches 111 are coupled to client communication paths 121, and one or more root caches 111 coupled to one or more server communication paths 131. Where appropriate, a set of intermediate caches 111 are coupled to the leaf caches 111 and to the root caches 111.

In a preferred embodiment, the graph structure is a tree structure, with a single root cache 111 and a plurality of leaf caches 111. For example, in a cache system

110 disposed for use with an ISP (internet service provider), there is one root cache 111 coupled to an internet backbone, and there is one leaf cache 111 for each POP (point of presence). In this example, the inter-cache communication path 112 includes direct connections (such as T1 or T3 connections) between the root cache 111 and each leaf cache 111.

Cache Devices

Each cache 111 includes a processor, program and data memory, and memory or storage 112 for recording one or more web objects 133. Each cache 111 retains the web objects 133 for repeated serving to client devices 120 in response to web requests.

In a preferred embodiment, each cache 111 includes a router-switch 113, for receiving messages and distinguishing types of messages that should be processed by the cache 111 from those that should not. For example, the router-switch 113 can divert all requests using FTP (file transfer protocol) or HTTP (hypertext transfer protocol) to the cache 111 for processing, while passing through other types of requests unchanged.

In a preferred embodiment, each cache 111 includes a cache device such as described in the Cache Disclosures, hereby incorporated by reference as if fully set forth therein, and is disposed for operating as described therein.

1 Multiple Cache Communication

2

3

4

Each leaf cache 111 receives requests from client devices 120 for web objects 133. The web objects 133 might be cacheable or non-cacheable.

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

If a client device 120 requests a cacheable web object 133, the leaf cache 111 might already have the requested web object 133 in its memory or storage 112. If so, the leaf cache 111 serves the requested web object 133 to the client device 120 without having to request the web object 133 from the root cache 111 or from the server device 130. If the leaf cache 111 does not already have the requested web object 133, the leaf cache 111 requests it from the root cache 111.

The root cache 111 performs a similar caching function, returning the requested cacheable web object 133 directly to the leaf cache 111 if it is already present in its own memory or storage 112, without having to request that web object 133 from the server device 130. If the root cache 111 does not already have the requested web object 133 in its memory or storage 112, the root cache 111 requests it from the server device 120.

If the leaf cache 111 and the root cache 111 do not already have a copy of the web object 133 in their respective memory or storage 112, the root cache 111 requests the web object 133 from the server device 120. Similarly, if the web object 133 is con-

1. sidered not cacheable, the root cache 111 requests the web object 133 from the server de-
2 vice 120 whether or not it has already that web object 133 in their respective memory or
3 storage 112. The server device 120 receives the request and returns the requested web
4 object 133 to the root cache 111.

6 Objects Already in Storage

8 The root cache 111 receives the requested web object 133 from the server
9 device 110, records it in its memory or storage 112, and determines an object signature
10 134 for the web object 133. In a preferred embodiment, the root cache 111 computes the
11 object signature 134 itself. In alternative embodiments, the server device 120 may com-
12 pute and record the object signature 134 and transmit it to the root cache 111 with the
13 web object 133.

15 In a preferred embodiment, the object signature 134 includes an MD5 digest
16 of the web object 133. In alternative embodiments, the object signature 134 may com-
17 prise a CRC, MD4, SHA, or other known function of the web object 133.

19 There is no particular need for any device to be able to recover the web ob-
20 ject 133 a priori from the object signature 134. It is sufficient that the root cache 111 or
21 the leaf cache 111 can determine, in response to the object signature 134, if the web ob-

1 ject 133 is present in its memory or storage 112, and if so, which web object 133 corre-
2 sponds to that object signature 134.

3
4 If the web object 133 is cacheable but was requested from the server device
5 110, the request from the server device 120 was due to a cache miss. However, it can still
6 occur that the leaf cache 111 (or some intermediate cache 111) already has the web ob-
7 jects 133 in its memory or storage 112, such as recorded in association with a different
8 URL (uniform resource locator) or other identifier. In a preferred embodiment, each
9 cache 111 records web objects 133 in association with the URL used to request those web
10 objects 133.

11
12 For a first example, multiple server devices 120 can record mirror copies of
13 identical web objects 133. For a second example, non-identical web objects 133 can in-
14 clude identical embedded web objects 133 (such as common graphics, animation, or pro-
15 gram fragments).

16
17 If the web object 133 is considered non-cacheable, it was requested from
18 the server device 120 because non-cacheable web objects 133 are not meant to be served
19 from the cache 111. However, it can still occur that the leaf cache 111 (or some interme-
20 diate cache 111) already has the web objects 133 in its memory or storage 112, because
21 the non-cacheable web object 133 had been requested earlier.

22

1 For a first example, if the web object 133 is responsive to a CGI script or
2 database search, it can be identical to the results of an earlier response to that CGI script
3 or database search. For a second example, if the web object 133 is determined dynami-
4 cally by the server device 130 (such as randomly selected advertisements), it can be iden-
5 tical to an earlier advertisement transmitted by the server device 130.

6
7 The root cache 111 transmits the object signature 134 to the leaf cache 111.
8 The leaf cache 111 determines, in response to the object signature 134, whether it already
9 has the associated web object 133 in its memory or storage 112 and if so, which one is the
10 associated web object 133. If so, the leaf cache 111 serves the associated web object 133
11 to the client device 120 from its memory or storage 112 without the root cache 111 hav-
12 ing to actually transmit the entire web object 133. If not, the root cache 111 transmits the
13 actual web object 133 to the leaf cache 111, which can then serve it to the client device
14 120.

15
16 In a preferred embodiment, the root cache 111 includes a bitmap 114 in its
17 memory or storage 112 for each non-cacheable web object 133, including one bit 115 for
18 each leaf cache 111. Each bit 115 of the bitmap 114 indicates whether its associated leaf
19 cache 111 has a copy of the web object 133.

20
21 The root cache 111 directly transmits the actual web object 133 to the leaf
22 cache 111 if the associated bit 115 of the bitmap 114 indicates that the leaf cache 111

1 does not have the web object 133. If the bit 115 indicates that the leaf cache 111 does
2 have the web object 133, the root cache 111 attempts to transmit only the object signature
3 134. However, even if the bit 115 indicates that the leaf cache 111 does have the web
4 object 133, it may occur that the leaf cache 111, being a cache, has discarded the web
5 object 133 in the interim. In this case, the leaf cache 111 so indicates and re-requests the
6 web object 133 from the root cache 111.

7
8 In a preferred embodiment, when the root cache 111 transmits the object
9 signature 134 to the leaf cache 111, it so indicates using a data type, such as a MIME
10 type, or a new type of object, indicating that the transmission includes only the object sig-
11 nature 134.

12 13 Compression for Transmission

14
15 When transmitting actual web objects 133 between caches 111 (such as
16 from the root cache 111 to the leaf cache 111), those web objects 133 are substantially
17 compressed for transmission and decompressed after reception. Compression for trans-
18 mission can be applied both to cacheable and to non-cacheable web objects 133.

19
20 Compression for transmission can include various techniques, such as
21 Huffman coding, Liv-Zempel compression, or other known lossless compression. Com-
22 pression for transmission can also include known lossy compression, such as JPEG,

1 MPEG, or other audio and video codec techniques, when appropriate for the type of web
2 object 133.

3

4 Those skilled in the art will recognize, after perusal of this application, that
5 transmission of the object signature 134 in place of the actual web object 133 is a form of
6 substantial compression. This form of compression is unreliable, in the computer science
7 sense that the receiver is not guaranteed to be able to recover the web object 133 from its
8 object signature 134. In fact, using this form of compression the leaf cache 111 can only
9 do so if the web object 133 is already recorded in its memory or storage 112.

10

11 Unreliable Dictionary Compression

12

13 As used herein, "dictionary compression" means a form of communication
14 in which a sender and a destination each maintain a set of dictionary elements and a set of
15 associated tag values, each tag value being representative of one of the dictionary ele-
16 ments. There is no particular requirement that the dictionary elements can be recovered
17 from their associated tag values without further information. Rather, dictionary compres-
18 sion refers generally to a system in which the dictionary elements can be associated with
19 arbitrary tag values.

20

21 The sender and the destination each associate the same tag value with the
22 same dictionary element. For example, the sender can transmit the dictionary element,

1 along with an arbitrarily selected tag value, to the destination to make the association.
2 Systems in which the sender does this, and the destination maintains a dictionary of such
3 tag values in response thereto, are known in the art.

4
5 As used herein, "unreliable" dictionary compression means that the desti-
6 nation might possibly discard the association between the tag value and the dictionary
7 element.

8
9 In a preferred embodiment, each dictionary element includes a complete
10 web object 133, and the tag value associated with each particular web object 133 is a
11 known function of that particular web object 133. The known function is preferably an
12 MD5 signature, as noted herein.

13
14 In a preferred embodiment, the destination (because it is a cache) can dis-
15 card any particular web object 133, and thus lose the association between that particular
16 web object 133 and its MD5 signature. That is, the destination (because it has discarded
17 the particular web object 133) can no longer determine if a particular MD5 signature is
18 associated with any known web object 133. Moreover, the destination cannot determine
19 the web object 133 in response to the MD5 signature without further information.

20
21 Transmission of the object signature 134 in place of the actual web object
22 133 is a form of dictionary compression in which the entire actual web object 133 is the

1 dictionary element. If the leaf cache 111 has discarded that dictionary element, it requests
2 the root cache 111 to retransmit the actual web object 133 using a second form of com-
3 pression. For example, the second form of compression can include a known lossless
4 compression technique such as Liv-Zempel compression or the form of compression used
5 in the PKZIP product available from PKWare, Inc.

6

7 Those skilled in the art will recognize, after perusal of this application, that
8 unreliable dictionary compression is applicable in various other applications that can use
9 compression. In a preferred embodiment, unreliable compression is acceptable because
10 the root cache 111 can retransmit the web object 133 using a more reliable (but possibly
11 less strong) compression technique.

12

13 Other Web Object Information

14

15 The root caches 111 and the leaf caches 111 can also exchange other infor-
16 mation about the web objects 133.

17

18 In a preferred embodiment, the cache system 110 collectively maintains in-
19 formation for each web object 133 regarding the following:

20

1 o A probability the web object 133 in the cache system 110 will be next requested by
2 some client device 120. This information will likely be best available at the leaf
3 caches 111.

4
5 and

6 o A probability the web object 133 in the cache system 110 will be stale. This in-
7 formation will likely be best available at the root caches 111.

8
9 The cache system 110 can collectively determine from this information
10 whether the web object 133 is the next web object 133 recorded by the cache system 110
11 to be served state. As described in the Cache Disclosures, particularly attorney docket
12 numbers CASH-003 and CASH-004, this information can be used to determine which
13 web objects 133 to actively refresh.

14
15 Active refresh can also be applied to frequently-requested non-cacheable
16 web objects 133, and distributed within the cache system 110, even though those web
17 objects 133 are re-requested from the server devices 120 each time. Active refresh is well
18 suited to web objects 133 such as advertisements, news reports, stock quotes, weather re-
19 ports, and the like.

20
21 The cache system 110 can also maintain information about each web object
22 133 regarding at which cache 111 in the cache system 110 that web object 133 is re-

1 corded. With this information, the root cache 111 can request cached web objects 133
2 from one of the leaf caches 111, in addition to or instead of re-requesting the web objects
3 133 from server devices 120.

4 5 *Method of Operation*

6
7 Figure 2 shows a process flow diagram for a method of using a system
8 having multiple caches.

9
10 A method 200 is performed by the system 100, including the cache system
11 110, the client devices 120, and the server devices 130.

12
13 At a flow point 210, one of the client devices 120 is ready to request a web
14 object 133.

15
16 At a step 211, one of the client devices 120 sends a message to its associ-
17 ated leaf cache 111 requesting a selected web object 133. The request message preferably
18 uses the FTP or HTTP protocol, and includes a URL for the selected web object 133.

19
20 At a step 212, the leaf cache 111 determines if the web object 133 is cache-
21 able or non-cacheable. If the web object 133 is cacheable, the method 200 proceeds with

1 the next step. If the web object 133 is non-cacheable, the method 200 proceeds with the
2 flow point 220.
3

4 At a step 213, the leaf cache 111 determines if the web object 133 is present
5 in its memory or storage 112. In a preferred embodiment, the leaf cache 111 makes this
6 determination in response to the URL for the selected web object 133 included in the re-
7 quest from the client device 120. If the web object 133 is present, the method 200 pro-
8 ceeds with the next step. If the web object 133 is not present, the method 200 proceeds
9 with the flow point 220.
10

11 At a step 214, the leaf cache 111 serves the web object 133 to the client de-
12 vice 120. The method 200 continues with the flow point 210.
13

14 At a flow point 220, the leaf cache 111 is unable to serve the web object
15 133 from its memory or storage 112, either because there has been a leaf cache miss or
16 because the web object 133 is non-cacheable.
17

18 At a step 221, similar to the step 211, the leaf cache 111 sends a message to
19 the root cache 111 requesting the web object 133.
20

21 At a step 222, similar to the step 212, the root cache 111 determines if the
22 web object 133 is cacheable or non-cacheable. If the web object 133 is cacheable, the

1 method 200 proceeds with the next step. If the web object 133 is non-cacheable, the
2 method 200 proceeds with the flow point 230.

3

4 At a step 223, similar to the step 213, the root cache 111 determines if the
5 web object 133 is present in its memory or storage 112. In a preferred embodiment, the
6 root cache 111 makes this determination in response to the URL for the selected web ob-
7 ject 133 included in the request from the client device 120. If the web object 133 is pres-
8 ent, the method 200 proceeds with the next step. If the web object 133 is not present, the
9 method 200 proceeds with the flow point 230.

10

11 At a step 224, similar to the step 214, the root cache 111 transmits the web
12 object 133 to the leaf cache 111. The method 200 continues with the flow point 210.

13

14 At a flow point 230, the root cache 111 is unable to transmit the web object
15 133 from its memory or storage 112, either because there has been a root cache miss or
16 because the web object 133 is non-cacheable.

17

18 At a step 231, similar to the step 211, the root cache 111 sends a message to
19 the indicated server device 130 requesting the web object 133. The request message pref-
20 erably uses the FTP or HTTP protocol, and includes a URL for the selected web object
21 133.

22

1 At a step 232, the server device 130 transmits the web object 133 to the root
2 cache 111.

3
4 At a step 233, the root cache 111 determines an object signature 134 for the
5 web object 133.

6
7 At a step 234, the root cache 111 determines if the web object 133 is present
8 in its memory or storage 112. In a preferred embodiment, the root cache 111 makes this
9 determination in response to the object signature 134. If the web object 133 is present,
10 the method 200 proceeds with the next step. If the web object 133 is not present, the
11 method 200 proceeds with the flow point 240.

12
13 At a step 235, the root cache 111 determines if the web object 133 is likely
14 present at the requesting leaf cache 111. In a preferred embodiment, the root cache 111
15 makes this determination in response to the bitmap 114 for the web object 133. If the
16 web object 133 is likely present at the leaf cache 111, the method 200 proceeds with the
17 next step. If the web object 133 is likely not present at the leaf cache 111, the method
18 proceeds with the flow point 240.

19
20 At a step 236, the root cache 111 transmits the object signature 134 to the
21 leaf cache 111.

1 At a step 237, the leaf cache 111 determines if the web object 133 is present
2 in its memory or storage 112, in response to the object signature 134. If the web object
3 133 is not present, the method 200 proceeds with the next step. If the web object 133 is
4 present, the method 200 proceeds with the flow point 240.

5
6 At a step 238, the leaf cache 111 transmits a message to the root cache 111
7 indicating that the web object 133 is not present.

8
9 At a step 239, the root cache 111 transmits the actual web object 133 to the
10 leaf cache 111. As noted above, the actual web object 133 is compressed for transmission
11 and decompressed upon reception.

12
13 At a flow point 240, the leaf cache 111 is ready to serve the web object 133
14 to the requesting client device 120. The method proceeds with the step 214.

15
16 *Alternative Embodiments*

17
18 Although preferred embodiments are disclosed herein, many variations are
19 possible which remain within the concept, scope, and spirit of the invention, and these
20 variations would become clear to those skilled in the art after perusal of this application.